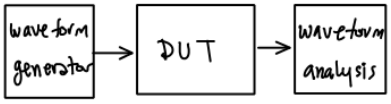
Lab 1 - Generating Test Signals

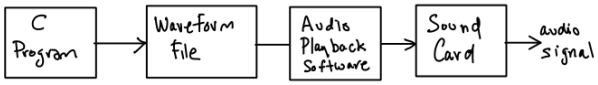
Modified 2013-9-13 to correct the bit pattern for an 8-bit signed number with value -128 (it's 10000000 not 11111111).

Introduction

Testing communication systems often requires generating test signals (waveforms) that are input to a system or component in order to see how it responds. The component being tested is often called the "Device Under Test" (DUT).



In this lab you will write a C program that generates a file containing a digitized test waveform. Then you will "play" these samples out through a digital-to-analog (D/A) converter (DAC) to create the test signal. You will observe the analog test signal on an oscilloscope.



The technique demonstrated in this lab is very useful because it can be used to test any device or system, even those for which no specialized test equipment exists.

In this lab we will be using a PC's sound card as the D/A converter. This works well for audio-frequency signals. For higher frequencies a specialized piece of test equipment called an Arbitrary Waveform Generator (AWG) is used.

Procedure

Use the Notepad program (under `All Programs -> Accessories-> Notepad`) to create and edit the C program file. Save the file on your own network drive (typically `H:`) so that you don't lose it when you log out of the lab PC.

Start a command prompt (also known as a command interpreter or "shell") (`All Programs -> Accessories-> Command Prompt`).

Type the command `cd` *folder-name* to Change Directory to the drive and folder (directory) where your C file is stored.

Type the command: `tcc -run` *your-file-name*`.c >lab1.out`) to compile and run your program using the Tiny C Compiler (`tcc`). The `>` character redirects the standard output of your program to the file name given after `>` (`lab1.out` in this example). If there are any error messages, correct the errors and re-run tcc.

Audacity is an audio editor that can also output waveforms through the PC's audio output DAC. Run `All Programs -> Audacity` to start the program. Since your waveform file is not in any known audio file format, use `File -> Import -> Raw Data` to read the waveform sample values from the file. You will have to specify the file and format. Choose the same format, number of channels and sampling rate as you assumed when creating the audio file (see below).

Check the displayed waveform to make sure the waveform polarity and values are correct. Take a screen capture that shows the time scale, the sample rate, sample format and the waveform (use `All Programs -> Accessories -> Snipping Tool`) and save it to your `H:` drive for use in your report.

Select `Transport -> Loop Play` to repetitively output the file to the PC's audio output.

Connect an audio cable to the headphone output on the front of the computer case. Clip the 'scope probe to the contact on the end of the plug on the other end of the cable. Adjust the 'scope gain, sweep and trigger to view the waveform. Have the instructor check and record that your waveform as displayed on the 'scope is correct.

Waveform Specifications

Your C program should output a waveform composed of a sequence of pulses, each pulse having a duration of 1 ms.

Each pulse should have a level of either the maximum or the minimum allowed sample value. The actual output voltages can't be determined since the audio DAC output levels and and amplifier gain are not calibrated. For 8-bit signed values the maximum value is +127 and the minimum value is -128.

Your program must generate pulses that encode a 7-bit binary number that is the sum of all the digits of your student ID. For example, if your ID number is A00123456 the number you would output is 21 (binary 0010101). This number may or may not match that of anyone else doing the lab.

The value should be output in little-endian order (least significant bit first). A positive pulse should be used to represent a '0' and a negative pulse for a '1'. This is a common encoding although both the bit order and the signal polarity are the opposite of what you might expect. For the above example you would output pulses with polarities `-+-+-++`).

Output an extra high pulse (equivalent to an extra '0' value) before the pulses corresponding to your number to make it possible to determine the start of the pulse sequence. Output four trailing low pulses (equivalent to four extra '1' values) after your number so you can easily see where the pulse train ends. Thus the waveform you write to the file should consist of a total of 12 pulses (a 'start' bit, seven data bits and four 'stop' bits).

Assume a sampling rate of 8 kHz. This is a common sample rate for speech signals since they are usually intelligible using only frequency components below 4kHz. Calculate the corresponding sample period and the number of samples you need to output to create each 1 ms-long pulse.

Hints

- `#include <stdio.h>` and use `putc()` or `putchar()` to write bytes to the standard output.
- declare the variables you write as 'signed char' if you want the byte values to be encoded in signed two's-complement notation (-128 being encoded as a byte with its bits set to 10000000 and +127 being 01111111).
- use a for loop (possibly within a function) to output the required number of samples per pulse

Pre-Lab

- compute the sum of the digits in your student number. Check your result.
- compute the sample period and number of samples per pulse.
- write a C program that generates the required waveform *before* the lab. Save the program on your H: drive or bring it to the lab on a "flash" drive.

Post-Lab Report

- submit a report that includes:
- your name and BCIT ID, course, date and lab number
- the calculation of your unique number in decimal and the conversion to bits and voltage levels. Use a table showing the pulse number from 1 to 12, the bit value and the signal polarity.
- the Audacity screen capture
- the program listing with all errors corrected
- export or print the report to a PDF format file and submit it via the course web site dropbox for Lab 1 reports